



BOSCH
Invented for life

Infotainment and Navigation Test Monitor (INTM)

Team3:

Danielle Valerie Guir
Scott Hansen
Joe Jiang
Jason Ostroski
Rudina Alhamzi

Executive Summary:

The infotainment system developed at Bosch's Car Multimedia Division is aimed to revolutionize driver safety and driving experience by providing a hands-on-wheel experience at all times method. In order to ensure all entertainment and navigation functions are successfully operating and to further improve the system, extensive on-road testing must take place to identify software bugs and ultimately contribute a flawless product. The testing can range between a short ride to the gas station to crossing multiple states, and possible errors can be software related or errors created as a result of user input.

This project was assigned to our design team with the goal of creating a monitoring system with video and audio recording capabilities. This system will be able to observe and monitor the GPS turn-by-turn directions and display screen to ensure the information given to the driver is accurate and consistent with the available routes on the road.

The current solution that Bosch utilize requires multiple devices, each capable of flagging for errors to recorded data. Flagging errors allowed for audio commentary to be recorded, as well. During on-road tests, this method required multiple devices and multiple riders, to flag issues synchronously and manually. This leaves room for human errors and is an expensive solution capable of recording a single view, since each device can only handle one camera. Our team was tasked to develop a solution as a suitable replacement to this current monitoring method, working with a reduced budget, approximately one third of their actual spending.

Given the time and resources, the team managed to develop a device with 4 Input cameras capable of recording data, and an output screen displaying four different views in one grid. Also, we have fulfilled Bosch's main desired device features:

- Record 4 videos of 4 different view in one grid window/frame
- Record one audio source
- save and replay data
- time and GPS stamping

Acknowledgment:

To Mr. Clark Smalley and Mr. Martin Beeker, for taking time out of your busy weeks to meet with us, once, every week, we would like to express our extreme gratitude. With your guidance and support we were able to complete our project up to this point. We are extremely grateful for the support and the advice along the way.

To Dr. Hayder Radha, our faculty facilitator, thank you for the weekly check-ins and for the support and guidance. We especially appreciated your input and advice, which helped and encouraged us all the way to the completion of this term.

Table of Contents

Executive Summary	1
Acknowledgments.....	2
Chapter1: Introduction and Background	4
Bosh infotainment system background.....	4
INTMConcept.....	5
Objectives, specifications and solution.....	5
Chapter2: Exploring the solution space and selecting a specific approach	6
Chapter3: Technical description of work performed	13
Chapter 4: Test data with proof of functional design.....	18
Chapter 5: Final cost, schedule, summary and conclusion.....	20
Appendix1: Technical roles, responsibilities, and work accomplished..	21
Appendix 2: Literature and website references	25
Appendix 3 and beyond (detailed technical attachments)	27

Chapter1: Introduction and Background

At the beginning of this semester, we were provided a document with our Sponsors brief description of the project and its features. After discussing and meeting with the sponsor, we came to clear realization of the particular features and purpose of making this device that we will explain in the following two sections.

Bosh infotainment system background

Bosch are known for their outstanding accomplishments in research and development of the audio entertainment systems from as early as 1932. During that year, they manufactured Europe's first car radio. The initial design was very heavy and expensive. However, Bosch's motto is "Invented for life"; they strive to design products that conveniently fit our daily lives. After 20 years from the first invented car radio, Bosch managed to design the first FM radio followed by the first transistor radio in 1957 that reduced the device's size. One of the interesting developments, is the Wasteland car radio in 1960. This car radio was fitted in the car, however it was portable and you could take it out and use it as a regular radio.

The pattern of inventions and developments in the decades after 1960 was continuous. from creating the car cassette player in 1965 to a quartz tuning system in 1979, a car CD player in 1985, and a Radio phone in 1997 to a globally adaptive and flexible infotainment system in 2010. Throughout these years, Bosch developed many modules as observed in figure 1.

1932	1949	1952	1957	1960	1965
First car radio, the Autosuper 5 (AS 5)	5A649, the first post-war car radio, with two frequency ranges	FM car radio	"Köln," "Bremen," "Hamburg," and "Berlin" car radios with transistor technology	"Westerland" combined car radio and portable travel radio	Car cassette player
1969	1974	1976	1979	1982	1985
"Frankfurt" stereo radio	ARI traffic news decoder	Car radio with integrated ARI traffic news decoder	Quartz tuning system, PLL synthesizer tuner	EVA electronic pilot for drivers	CD player
1986	1987	1988	1989	1992	1995
Key code for theft deterrence	Dolby C noise suppression, removable front panel	RDS radio data system	TravelPilot IDS, first vehicle navigation system in Europe	TIM traffic information memory	TravelPilot RGS05, navigation system with GPS control, route guidance, and speech output
1997	2000	2002	2003	2005	2008
Radiophone, a combined car radio and GSM cellphone	Dallas RDM 169 car radio with mini-disc drive	TravelPilot navigation system with online connection	Woodstock DAB53 digital car radio with MP3 drive and digital recorder	First radio navigation system with map zoom function and color display	First mobile navigation system with integrated camera for video navigation and recognition of speed-limit signs

Figure1.1 - Bosch's car Entertainment System Milestone

INTM Concept:

This year, Bosch has won the "Best-of-CES Award 2013 or Chevrolet My Link System" for its current Infotainment system. This particular user friendly system combines audio, video, entertainment, communication, and navigation systems. This system stands out from the other products in the market due to its options and features that can be expanded further by simply adding applications similarly to a smartphone. Also, it is an extremely user friendly system that uses a natural human voice input without having to memorize particular commands, providing a safer "hands on wheels at all times" driving experience. (Bosch)

In order to maintain the product's quality and monitor all possible errors in developing this award winning technology, there are different methods and devices that could be utilized. However, the only system that is currently available at Bosch's Car Multimedia Division for in-car, on-road, quality testing, is a fairly expensive set of video recording equipment. It is used to monitor and view the incoming data from the testing conducted on their Infotainment system.

In this available system, only one camera input is available per device. Also, to track any occurring errors during a road testing, testers must manually set flags. In order to test two different views, testers must use two devices. During the road test, there must be at least two (2) members in the car, who are synchronously setting flags via a button to start and stop the system. In addition, data recording whether it was written and/or via the device is also a manual operation. As the testing is done while on the road, this leaves a considerable amount of room for human error.

This current error-monitoring method also creates additional distraction for the driver from the road. Following the road test, video recorded data must be cut and edited by hand by one of the members of the team, to sort out essential and nonessential video clips, which can be tedious and time consuming.

Objectives, specifications and solution:

Having observed all these issues associate with the current solution, the Infotainment Development Bosch Team came up with a more cost and time effective solution that was handed to our team for development. Given the project's description and specifications, our team's objective is to develop a device that is capable of recording, compressing, and compiling 4 videos into one view. It must be able to monitor the GPS location of the device, maintain a reliable time stamp, and record audio. All collected data will be exported to a USB drive, where testers can view the recorded videos via laptop or PC without manually editing.

After brainstorming many ideas and solutions, the team members shared a vision of a fully automated, simple and user friendly solution. In the final deliverable, we have used the ODROID-XU microcomputer. Operation requirements are as follows:

- Connect 4 webcams via USB to ODROID-XU.
- Connect the numberpad via USB to ODROID-XU.

-Connect a power supply DC/AC adapter to ODROID-XU.

Overall, all team members agree that the solution they arrived at is cost efficient and consistent of easily attainable components and supported open source software. At Bosch car multimedia center, the current solution they have is costing them about approximately \$1400 per view (one device). Our solutions provides access to 4 different views at the same time for approximately \$500. the team successfully managed to cut cost by 10 times less than the current device they are using (assuming they need four different views). we expected our prototype to cost so much more, because we wanted to use more expensive components. however, we chose to trade some of the unnecessary features for price, while maintaining the quality.

Chapter 2: Development of a Solution

Based off the project descriptions and requirements, our team came up with a solution in one device. The device is expected to be capable of recording, compiling, and compressing multiple video streams and audio into one packaged deliverable. To confirm the accuracy of the Infotainment System's navigation features, the solution proposed by our team includes a GPS location of the device and was expected to keep a reliable timestamp. In addition to all of this, the solution was expected to have a way of flagging recorded data to isolate the meaningful data. All of this information was expected to be saved to a USB drive, in this way the meaningful data could be exported to a laptop or PC for use.

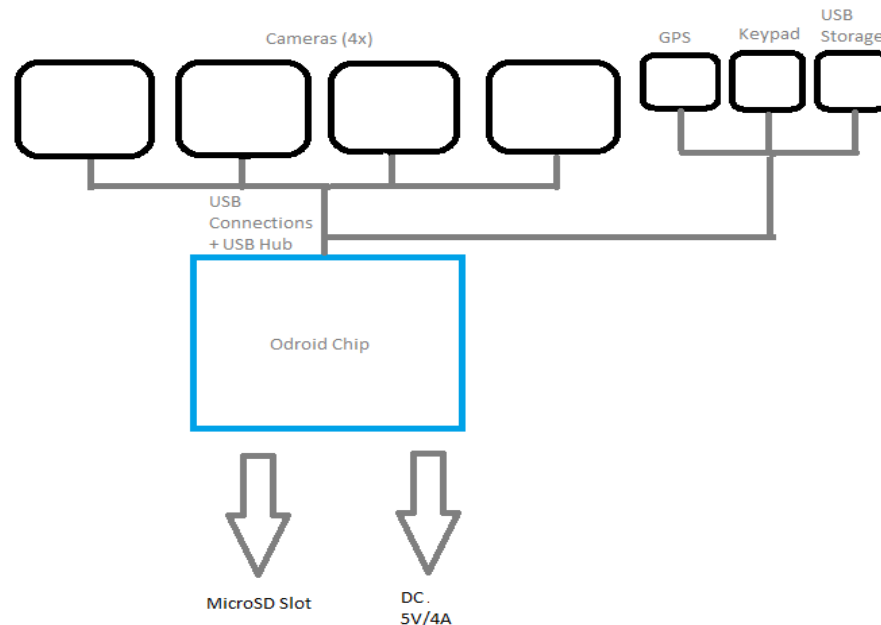
To go into further detail, the system described required four different cameras placed within vehicle. Two cameras focused on areas inside the vehicle; specifically, on the instrument cluster, located just behind the steering wheel, and one to monitor the center stack display, located on the main dashboard. These two would be intended to monitor and detect any errors occurring on the displays, as a result of software or as a result of human error. Two other cameras were intended to be placed on the dashboard facing outside the vehicle. These cameras were intended as a way to see what the driver is seeing and as a second way of confirming the information on the display - more specifically navigational information and instructions.

The data recorded by these cameras are intended to document any GPS, software, or human errors made resulting in system errors, as part of the testing process. Therefore, the cameras would expected to be recording simultaneously while the testing is taking place. Aside from the video being recorded, the system was also expected to have a way to monitor audio issues that may arise during the on-road testing.

Synced according to the timestamp, the solution also included a GPS position tracker to record the vehicle's location and displayed on the recorded video data. For data mobility, all of the information accumulated during the on-road test was expected to be saved on a storage device. It was decided that a USB drive would be the optimal medium used to save this data, as it could be removed from the device and would be compatible with most computers, though this also meant that the data must be compressed into a manageable file size. As this was apart of the current monitoring solution used by our customer, it was also agreed that for process assimilation, this method would allow for a more seamless implementation of our solution.

As a the workhorse of the solution, it was agreed that a microcontroller or microcontrollers be used. As part of the requirements, this or these microcontroller system(s) should be able to have the processing power to accomplish what was listed above and be able to operate on low power. For a flagging system, our team decided that a single button to flag the system, based on the timestamp, would be acceptable. For post processing, videos were expected to compile all four streams of recorded video data, the audio data, the GPS text data, and the time stamping information and compile them into one file, which would then be saved and exported, using the USB described above.

Together, the external design is as such; a main box containing the microcontroller and a collection of peripheral components. These peripheral components, consisting of the four web cameras, the USB GPS, keypad, and USB storage solution, would be connected either directly to the microcontroller or connected to a USB hub. To clarify, below is a graphical representation of our physical design:



Figur2.1- Basic Physical Design

This design solution left a considerable amount of room as to the specific hardware components that could be used. The Decision Solution Matrix (**Figure 3.2.1**) can summarize the comparison matrix which was instrumental in the our hardware decision. It is quite clear that our second design solution was the best option for the project, based on the customer requirements and availability. The Odroid gave us a smaller design, great processing power, and many USB ports including both USB 2.0 and 3.0 versions.

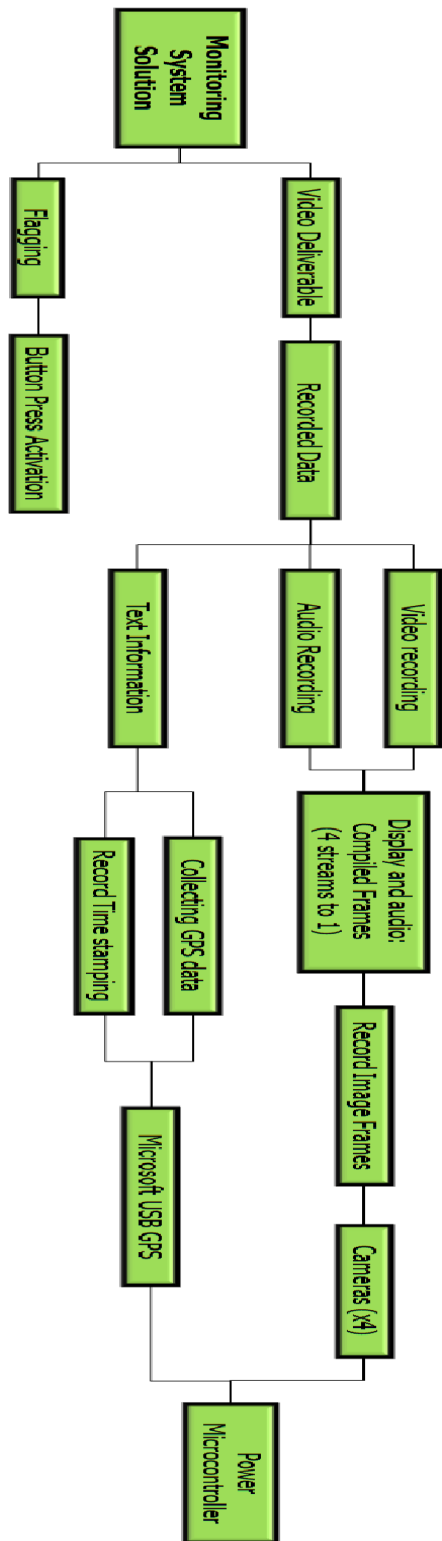
In addition, this was considered a reasonable solution as, provided the correct software, the solution would meet our financial limitations of \$500 and would be able to meet our customer's, Bosch's, requirements. The following information, also, aided in the decision to use the solution described above:

Design Parameters Importance Ranking

	4 Very Important	3 Important	2 Neutral	1 Least Important
Correct Camera resolution	•			
Audio quality		•		
GPS location information		•		
System robustness	•			
Error Flagging		•		
Display screen with unit			•	
Power management		•		
Storage			•	

The chart above was helpful as it identified the features to enable prioritization, which had an effect on the software we chose, as well as, playing a part in the type of hardware used for the overall solution. Based off of the Design Parameters Importance Ranking chart above, we we were able to break down the system even further to outline the primary structures and their supporting structures:

FAST Diagram



Task management

For efficiency and to track our progress during the lifetime of the project, we created a Gantt chart, the results of which can be found as a reference in Appendix C, titled *Original Gantt Chart*. This original project timeline was created to have some flexibility, accounting for class, work, and possible interview schedules. Development and research portions of the project were given a large range to account for fluctuating schedules and availability. Though it did not include weekly meetings outright, these were implied in the timeline. Also included were the due dates for some of the more critical reports and presentations that were required as part of the project components.

Unfortunately, even given the flexibility of the projected timeline, there were unforeseen scheduling conflicts that led to the new Gantt Chart, which can be found in Appendix C as a reference, titled *Updated Gantt Chart*. The discrepancies, mainly due to updated due dates made in-class, critical project due dates for other classes, and general conflicting schedules caused by class obligations, including, but not limited to lab obligations. These components contributing to what is the updated chart were unaccounted for during the creation of the original chart. Note: As an attempt to make the exported version of both our Gantt Charts more readable, the contents of both pdfs have been separated into “Tracking” and “Non-tracking” versions.

Project Projected Budget

To take on this undertaking, the team came up with a projected budget including a list of the components considered pertinent to the success of the project. Ultimately, during testing we found that there were components that would not be included in the final deliverable, due to lack of time and scheduling issues during early development, but that being said, the breakdown of the original projected budget, including the quantity, cost, and contribution to the total, is as follows:

Part	Quantity	Cost	Total
Odriod-XU	1	\$169.00	\$169.00
Odroid RTC Battery	1	\$8.00	\$8.00
8 GB MicroSD Card	1	\$13.00	\$13.00
32 GB MicroSD Card w/ Adapters	1	\$23.99	\$23.99
Logitech C270 Camera	2	\$33.99	\$67.98
Logitech C310 Camera	2	\$44.99	\$89.98
Microsoft USB GPS	1	\$49.95	\$49.95
32 GB USB Drive	1	\$19.99	\$19.99
USB Number Pad	1	\$12.56	\$12.56
MicroHDMI - HDMI Cable	1	\$9.00	\$9.00
USB3.0 Micro-A to Standard-A Host Adapter Cable	1	\$10.00	\$10.00
USB Hub	1	\$6.99	\$6.99
Total Cost			\$480.44

Chapter 3: Technical description of work performed:

Hardware Design Efforts

Our hardware design initially was based on a laptop as our main computing unit. While a laptop would have allowed for easy camera calibration due to having an integrated screen and would have been powerful enough to handle 4 videos recording simultaneously, it was too large based on Bosch's requirements. Our next chosen computing unit was NetTop computer, a very small form factor desktop; however, this was also too large.

We then researched the Raspberry Pi, BeagleBone Black, and other similarly powered and sized computers, however, all our researched options were either underpowered or did not have enough USB ports for connecting our cameras and peripherals. While conducting a phone interview with Clark Smalley and Martin Beeker, our contacts at Bosch, we explained our problems with selecting a computing unit and Clark emailed us a link to the Odroid-XU. The Odroid-XU, with a 1.6 GHz Quad Core CPU, 2 GB of RAM, and 6 total USB ports, was powerful enough to handle our video recording requirements, and with a total size of 100 x 74 x 29 mm while in an enclosure, fit the size requirement (2.3.1).

A microSD card or EMMC Module is required in order to run the Odroid-XU and either of these is necessary to store the operating system, software, and data. While researching these options, we considered utilizing an eMMC module as it has higher performance than a microSD card, however, HardKernel, the Odroid's manufacturer, only offers eMMC Modules with an Android Operating System installed. Our team had no experience developing software for Android, there was not sufficient documentation on loading eMMC Modules with our intended Ubuntu operating system, therefore, our team decided on utilizing a microSD card.

We initially purchased a 32GB microSD card for our Ubuntu 13.04 operating system and an 8GB microSD card with Android pre installed from HardKernel in the event that we ran into issues loading Ubuntu on the 32GB microSD. Once we had installed Ubuntu 13.04 on the microSD card, we noticed that it was very slow and was not as stable as needed for our prototype. Upon comparing this to the Android microSD and researching this problem, we realized that the speed and class of our microSD card was not high enough to handle Ubuntu with the performance we required.

Our originally purchased Kingston 32GB microSDHC Class 4 has a 4 MB/s sustained read/write speed. Our new Transcend 32 GB microSDHC Class 10 UHS-1 has a sustained 90MB/s read speed and 25MB/s write speed. Once installing Ubuntu on the new microSD card, the operating system ran much more smoothly and quickly, ensuring that our slow operating system problem was due to our original, slow microSD card.

We originally purchased one Logitech C310 HD Webcam and one Logitech C270 HD Webcam. Even though we needed four webcams for our final design, we only purchased two in the beginning phase of our design to ensure they worked with our design and provided the quality that we needed. We ordered one more of each model once they passed our testing.

The two Logitech C270 HD Webcams are utilized to record video of the Infotainment System and the Instrument Cluster and are 3 Megapixels with a 720p resolution for clear quality. The two Logitech C310 HD Webcams are utilized to record video of the surroundings of the car, mainly street signs. These are 5 Megapixels with a 720p resolutions and the higher megapixel rating allows for better digital zoom if needed when Bosch reviews the videos. Also, these cameras have auto light correction, which may allow for better video quality in variable lighting conditions. Both microphone models have a built in microphone to record audio and are UVC Compatible which means they are plug and play with most versions of Linux.

We originally thought we would need to utilize a USB 3.0 Micro-A to Standard-A Host adapter cable in order to utilize the USB 3.0 OTG port, but once we tested our design realized it was not needed. We also originally planned to utilize the Odroid-XU RTC battery to keep the real time clock of the operating system running when the Odroid was not connected to power, but our Microsoft GPS provides a time with the GPS coordinates. We found it much easier to utilize this data as opposed to the operating system's clock. The remainder of our hardware, such as the USB Hub and microHDMI to HDMI cable, did not need to meet strict specifications, and therefore, was mostly chosen on low cost compared to other models.

Software Implementation

Our original software design was to be implemented utilizing components of GUVView and Zoneminder, two open source softwares for Ubuntu. GUVView is a simple interface for capturing and viewing videos from any UVC compatible device (2.3.2), and Zoneminder is a home security and surveillance software (2.3.3). Our intended design was to study the source code of both of these softwares and utilize what we learned to build our own software. GUVView was to be utilized for its video and audio recording and Zoneminder was to be used for compiling the 4 videos into one and adding the time stamp and GPS stamp to the video.

Our initial research of both of these source codes were that they were very complex and implemented many features we did not need. A bash script was created that was able to create multiple instances of GUVView to record video. However, there were problems with the stability of the instances. For example, the user interface was different with certain buttons and functions unavailable while running multiple instances. Additionally, GUVView crashed at random periods of the recording, which raised an issue of reliability.

During this time, we also compiled and ran Zoneminder, however, we quickly learned that it was not only complex to set up, but it also only streamed the video to a web browser. Since our project needed to output to a USB drive, we decided to research more options and found Snowmix.

Snowmix is an open source video mixer tool for mixing live and recorded audio and video feeds (2.3.4). It allows for overlay of text for our GPS and time stamps and can compile 4 videos together. We ran into many issues while attempting to compile the source code for Snowmix with most of the problems stemming from the GStreamer plugin.

While researching the GStreamer plugin, we found a separate GStreamer FFmpeg plugin that is based on the FFmpeg library. FFmpeg is a free software project for recording, converting, and streaming audio (2.3.5) and is under the GNU Lesser General Public License in which individuals or companies are allowed to utilize the source code to build their own software without needing to make their software open source. Upon further investigation of FFmpeg, we quickly learned that it is a simple, command line driven solution that is not only able to record audio and video, but is able to compile 4 prerecorded videos into one and overlay text. A decision was made to focus the group's effort towards FFmpeg instead of the other software options.

Installing FFmpeg onto the Odroid consisted of choosing between downloading the static version or compiling the program locally. It was discovered that downloading the static version from FFmpeg's website (2.3.5) resulted in an error on the Odroid. This error was due to the static build being incompatible with the version of Ubuntu that was installed on the Odroid. To solve this problem, FFmpeg was installed using the other method of downloading the source code and compiling the program locally. After compiling using the commands "configure" and "make", FFmpeg was successfully installed on the Odroid.

At this point, we were able to record 4 videos simultaneously and utilize the “filter complex” to post process the 4 videos into one video (2.3.6). However, it was found that for the subtitles to overlay on video, libass filtering needed to be enabled in FFmpeg. To accomplish this, the command “./configure --enable-libass” was typed in the folder that contained the FFmpeg source files. After typing this, FFmpeg needed to be recompiled for the changes to take effect by typing “make”.

During our testing of the 4 video recording, we noticed that occasionally one or more of the cameras would not record. In our script we use “/dev/video*”, with the star being a 0, 1, 2, or 3, to specify which camera is utilized for which recording instance. Running a “ls /dev/video*” command from the terminal would list all of the /dev/video instances were enabled on the operating system. We found through running this command, that occasionally the cameras would not choose the first 4 numbers of the dev/videos.

Upon more research we realized that the way the dev/videos are assigned is through the hardware identification numbers and the port that they are connected to (2.3.7). Since we purchased 2 of each model camera, we had 2 instances of 2 hardware devices having the same identification number. Furthermore, the 4 USB 2.0 ports on the Odroid are not individual ports, but rather an internal USB Hub. To fix this issue, we connect one of each model of camera to the USB 2.0 ports and one of each model camera to the USB Hub, connected into the USB 3.0 port. This corrected our problem as the USB 3.0 port is not on the same port as the USB 2.0 ports and each of the same model cameras were then connected to same USB port.

The audio was recorded by utilizing the ASLA (Advanced Linux Sound Architecture) command in our recording script and specifying a hardware input device (2.3.8). While this worked properly in our initial testing, further into our development, the ASLA audio recording stopped working properly. When this occurred, the whole recording process wouldn’t stop at the correct time, and we kept getting an ASLA buffer error. In order keep development, we reverted to a recording script that did not implement the audio recording. If we had more time for development, this would be one of the problem areas we would concentrate on.

The overall layout of the software is a bash script that starts four instances of FFmpeg that are recording at set increments of 30 minutes. While the video is recording, the script is also collecting GPS data in a separate file. After 30 minutes, the videos are stored to a USB drive and the script transitions into the subtitling phase. In this phase, the GPS text file is parsed by the script so only the important data such as UTC time, latitude and longitude are saved. After the data is parsed, the script converts the “.txt” file into a “.srt” file. The “.srt” file is then converted into the necessary “.ass” extension that is used for subtitling by FFmpeg. Once the subtitling is done, the script transitions into the post-processing phase. This phase takes the four separate videos and combines them into one video. Once the four videos are combined, the GPS “.ass” subtitles are then overlaid onto the videos and stored to the USB drive.

GPS information was gathered by using Microsoft Streets and Trips. This GPS module transferred data to the Odroid by USB port. In order to trigger the GPS to collect data, sudo and cat commands were used: “sudo cat /dev/ttyACM0 > GPS”. This command would take the

GPS information and put it into a text file labeled GPS. The raw data from the GPS was formatted in the standard GPS format. Research was done in order to understand the raw data (2.3.9).

```
$GPGGA,043200.00,4243.47153,N,08428.86020,W,1,04,1.86,280.4,M,-34.7,M,,*6C
```

```
$GPGSA,A,3,20,32,31,16,,,,,,,,,7.47,1.86,7.24*0E
```

```
$GPGSV,1,1,04,16,37,190,37,20,42,279,36,31,55,049,39,32,53,226,35*7D
```

```
$GPGLL,4243.47153,N,08428.86020,W,043200.00,A,A*71
```

```
$GPRMC,043201.00,A,4243.47148,N,08428.86034,W,0.225,353.08,211113,,,A*71
```

```
$GPVTG,353.08,T,,M,0.225,N,0.417,K,A*37
```

As highlighted above, “\$GPGGA” is the only line that is important when looking at GPS data. Only the first few elements separated by commas are important. These elements are formatted like so:

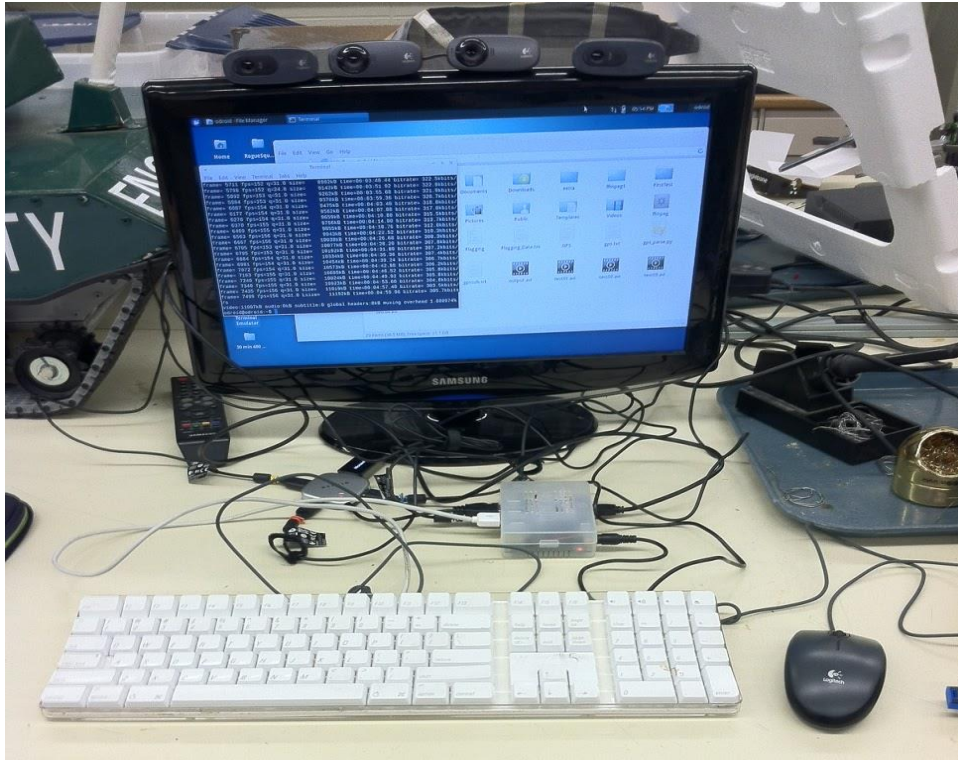
```
$GPGGA,043200.00,4243.47153,N,08428.86020,W
```

Message ID,UTC Time,Latitude,N/S Indicator,Longitude,E/W Indicator

A script in python was developed in order to parse and format this information. Python was chosen since it has many key commands that make parsing easier. For parsing, a for loop was created in order to search the raw data line by line. An if statement was put right after with a condition using the “line.startswith('\$GPGGA')” command. This command will take any line starting with \$GPGGA and run through the “line.split(',')” command. This command will separate all elements in the line into an array to complete the parsing process. For formatting, the necessary data was formatted by using the print command. The data is printed into the terminal. In order to export this data into a text file, we run the script in terminal with “python gps_parse.py > gpssub.srt”. The file can then be overlayed.

Flagging was done by pushing a button, that has a script hot keyed to run through the terminal. The hot keying was done by using a keyboard shortcut in Ubuntu’s setting (2.3.10). Once pressed the script will run the command “echo \$(date +%H\:%M\:%S).00 >> Flagging_Data.txt” which will take the current time on the real time clock of the Odroid and export it to the Flagging_Data.txt file.

Chapter 4: Test data with proof of functional design:



We tested our prototype in the ECE 480 lab utilizing a HDMI compatible monitor, mouse and keyboard, as shown in the photo above. We manually ran our created scripts for each portion of the recording and post processing process. While one of the requirements was to have our prototype be easily used without these peripherals, we were unable to complete this requirement due to time constraints. If we were to complete the design, it would have not utilized the monitor, mouse and keyboard, and would only need the USB number pad to stop and start recording. We tested the GPS by taking measurements with it the Engineering Courtyard.

Our completed design was tested in a car driving around Michigan State University's campus. The script was recording four videos for a 5 minute interval. The codec used was mjpeg and was recording at 15 frames per second with a resolution of 640x480. Increased quality can be used, however it greatly increases post-processing time. GPS data was then overlayed as subtitles on top of the four video output. A screenshot of our final test is shown below.



Chapter 5: Final cost, schedule, summary and conclusion (in the conclusion: what we would've done with more time + future features):

Our final prototype did implement many of the requirements specified by Bosch, but due to time constraints, did not implement all of the requirements. Recording 4 videos, compiling them into one video, GPS and time stamping, flagging, and outputting the video to a USB device to be reviewed on another computer were all completed. Although we would have liked the flagging to display on the actual video, we only could get it to output to a text file before our design was due. Audio recording, which did work initially, but failed in further development, was not implemented in our final prototype. Making the prototype run without the user needing to run scripts manually with a monitor, mouse, and keyboard is another requirement that was not met. However, this could have been implemented if given more time to develop.

Much of our software development was not utilized in the final prototype due to the fact that we found our final solution software, FFmpeg, so far into the development process. The flagging, audio, and automated scripts requirements could all be easily implemented by another design team, utilizing the hardware we chose and the scripts we created.

Our final prototype is under budget, at \$462.77 of our allotted \$500. This is less than our original estimate of \$480.44 as many of the parts were found at a lower price and Bosch sent us a Kingston 32 GB USB Drive.

PART	Quantity	Cost	Total
Odroid-Xu	1	\$169.00	\$169.00
Odroid RTC Battery	1	\$8.00	\$8.00
8GB Micro SD w/ Android Installed	1	\$13.00	\$13.00
Transcend 32 GB microSD HC Class 10 UHS-1	1	\$22.99	\$22.99
Kingston 32 GB microSD card HC class 4	1	\$26.99	\$26.99
Logitech C270 Camera	1	\$24.99	\$24.99
Logitech C270 Camera	1	\$32.23	\$32.23
Logitech C310 Camera	2	\$29.99	\$59.98
Microsoft USB GPS	1	\$47.04	\$47.04
USB Number Pad	1	\$12.56	\$12.56
MicroHDMI - HDMI Cable	1	\$9.00	\$9.00
Belkin USB 2.0 4-Port Hub	1	\$6.99	\$6.99
Total Shipping		\$30.00	\$30.00
Total Cost			\$462.77

Appendix 1 - Technical Roles and Contributions



Right to left: Jason Ostroski – Danielle Valerie Guir – Scott Hansen – Rudina Alhamzi- Joe Jiang

Jason Ostroski:

Jason worked both on the hardware and software portion of the design. He aided in selecting the hardware based on Bosch's requirements. He also worked on choosing the operating system and loading it onto the MicroSD card for the Odroid. Once the Ubuntu 13.04 operating system was finalized on the Odroid, Jason aided team members in loading the same operating system onto their personal machines and virtual machines in order to standardize development. He aided in the open source software choice and compiled them onto both his own laptop and the Odroid for evaluation. Furthermore, he realized hardware shortcomings when the Ubuntu 13.04 operating system was running slow on the Odroid and chose a faster class of microSD card to combat this issue. He worked on writing scripts for recording the video and audio, filtering it into 4 views, and storing the data. He also did extensive testing of our scripts on the Odroid and ensured that the scripts written on the individual team members laptop's worked properly on the Odroid. Jason also found the cause of problem in which one or more cameras would not record properly in the 4 video recording script. He researched this

problem and found a way to fix this problem without having to change our hardware or software design.

Danielle Valerie Guir:

Danielle Valerie contributed to both Hardware and Software portions of the project. She aided with the software and hardware solutions, based on the requirements set out by Bosch's contacts. From a hardware standpoint, apart of the discussion and decision process of choosing hardware components, which is especially true for the video and audio recording hardwares used for the project. For these, she referenced a number of online component sites, hardware reviews, and compared pricing and specifications. This led to the decision to use the Logitech C270 and C310 for the cameras used to focus in and out of the vehicle.

From a software standpoint, she was not only active in the research, decision, and architecture of the initial software design, but she was also active in participating in the development of the scripts that are currently being used in the final solution. As the software solution evolved from using GUVVIEW, Snowmix, Zoneminder, and then finally to FFMPEG, she contributed greatly as she accessed multiple sources to research on the various software solutions, scripting solutions, etc., and provided them to the individuals running test scripts on the actual system. She also set up the GitHub repository, which was instrumental earlier on in the project to maintain consistent software versions across testing environments. Her contribution was especially felt during the testing process of the software and implementation as it related, but was not limited, to the display and compression. This was also true for sections that related to the selection and debugging of the end-to-end solution that is being implemented in the solution currently.

Finally, she contributed to the website design and implementation, making updates to content and formatting so as to aid with user accessibility and content meaningfulness. This involved modifying the HTML and the CSS for a better and more meaningful user experience, as well as, edits to the JQuery.

Scott Hansen:

Scott contributed to the software design, team website, and certain aspects of the hardware. He helped to determine which pieces of hardware would best be compatible with the Odroid and how to implement them. Operating system selection and installation was also a contribution. He helped to mount the operating system through the use of Win32DiskImager that burned an image to the microSD card. Ubuntu 13.04 was selected as the Operating system of choice, and he downloaded the necessary packages that would support the applications that were used. He helped to determine software layout and feasibility between GUVView, Snowmix, Zoneminder, and FFmpeg. FFmpeg was chosen as the primary software choice, and he researched the necessary program commands and packages that would accomplish the goals for the project. Additionally, he contributed to writing scripts that would record four videos at once, overlay subtitles, and post-process the four videos into one. Furthermore, he helped to resolve issues that arose between configuration options of programs that would

cause conflicts on different operating systems. For example, the libass subtitle package worked on the team's local machines, however the program did not work when tested on the Odroid's operating system. After performing research, he resolved the issue by installing the program from source and compiling manually with customized configure flags. Additionally, he spent most of his time researching into the functions that would be necessary to perform each task. The four video capture required parameters that would set the time, frame rate, and resolution of each camera at startup. He also created the team website through the use of Dreamweaver and is located at <http://www.egr.msu.edu/classes/ece480/capstone/fall13/group03/>. The website has elements such as a JQuery slider for photos. Each team member wrote Application Notes about topics related to the project that are linked on the website.

Rudina Alhamzi:

As a team member and the Document prep, Rudina Alhamzi contributed in more than just that role. She also assisted in hardware and software progress of the Project. Regarding her Document prep roll, she always created the layout of our proposal, design Issue paper, progress reports and all other documentations. Also, she assured that sections were distributed equally between other team members aside to editing sections and confirming that they meet the writing criteria. Regarding the Hardware aspect of the project, she assisted in brainstorming and come up with different Ideas on how to power the ODROID-XU board we are using in a car, aside to some camera mounting solutions. Moreover, despite her lack of programming knowledge, she still attended every group meeting the team carried out. In the software aspect she assisted in searching for script solutions and implementing the features and testing them using a command line window.

Joe (Yizhou) Jiang:

Joe Jiang played a vital role in Hardware and Software development and integration. He helped determine the C270 and C310 webcams were the most cost efficient webcams for what our projects objective. Webcams were determined with analysis of cost, contrast, brightness auto correction, resolution, and ease of use. Besides playing a vital role in deciding specifying hardware specifications, he also completed research and development for key program features requested by Bosch. Key software features that he worked on consist of GPS/time stamping and error flagging. GPS hardware integration was done by doing research on how external gps devices interact with the microcontroller. Interactions such as triggering the device with terminal and how it exports data was key aspects to the project. Software development was done with the GPS raw data by learning python. He demonstrated his knowledge in python by creating an efficient program able to export a .srt file ready to overlay. Error flagging was done by creating a script to record clock data and hot keying it to make it user friendly. Hot keying required research on how to integrate the numpad and map function keys to it. Without the knowledge on remapping keys the hot keying will not work due to Ubuntu's constraints. On the software side of flagging, Joe demonstrated his ability to create batch scripts that can communicate with data from the real time clock of the microcontroller. After completing these tasks he expanded on his responsibilities and helped other team members in integrating his

programs into other peoples scripts and explaining the hardware integration. With his research and technical knowledge he was able to excel the completion of the project.

Appendix 2 – Literature and Website references

1.1.1

Bosch Automotive A product history." BOSCH, n.d. Web. 1 Dec. 2013.

<http://www.bosch.com/content2/publication_forms/en/downloads/Sonderheft_2_automotive_en.pdf>.

2.3.1

Odriod-XU. Hardkernel, n.d. Web. 04 Dec. 2013.

<http://hardkernel.com/main/products/prdt_info.php?g_code=G137510300620>.

2.3.2

Assis, Paulo. *GTK UVC Viewer*. Sourceforge, n.d. Web. 04 Dec. 2013.

<<http://guvcview.sourceforge.net/>>.

2.3.3

"ZoneMinder - ZoneMinder: Linux Home CCTV and Video Camera Security with Motion Detection." *ZoneMinder*. N.p., n.d. Web. 04 Dec. 2013. <<http://www.zoneminder.com/>>.

2.3.4

Snowmix. N.p., n.d. Web. 04 Dec. 2013. <<http://sourceforge.net/p/snowmix/wiki/Home/>>.

2.3.5

FFmpeg. N.p., n.d. Web. 04 Dec. 2013. <<http://www.ffmpeg.org/>>.

2.3.6

Create a Mosaic out of Several Input Videos – FFmpeg. N.p., n.d. Web. 04 Dec. 2013.

<https://trac.ffmpeg.org/wiki/Create_a_mosaic_out_of_several_input_videos>.

2.3.7

"How Can I Assign a Webcam to a Specific /dev/video#?" *Linux Questions*. N.p., n.d. Web. 04 Dec. 2013. <<http://www.linuxquestions.org/questions/linux-hardware-18/how-can-i-assign-a-webcam-to-a-specific-dev-video-669507/>>.

2.3.8

"FFmpeg Devices Documentation." *FFmpeg*. N.p., n.d. Web. 04 Dec. 2013.

<<http://www.ffmpeg.org/ffmpeg-devices.html>>.

2.3.9

. *Robot Shop*. N.p., n.d. Web. 4 Dec 2013. <<http://www.robotshop.com/media/files/PDF/user-guide-ktgps1.pdf>>.

2.3.10

. "How to link a custom keyboard shortcut to a bash script in Ubuntu 13.04?." *Ask Ubuntu*. N.p., July 30 2013. Web. 4 Dec 2013. <<http://askubuntu.com/questions/326282/how-to-link-a-custom-keyboard-shortcut-to-a-bash-script-in-ubuntu-13-04>>.

Appendix 3 – Additional Technical attachments

Figure 3.2.1 - Decision Solution Matrix

Engineering Criteria	Importance (Score: 1-5)	Possible Solutions			Reasoning - Customer Requirements perspective
	5=High, 1=Low	Solution #1	Solution #2	Solution #3	
Hardware Platform		Asus NetTop	Odroid	Raspberry Pi	
CPU	5	Intel Atom D2700 (2.13GHz, dual core)	EXynos5 Octa Cortex - A15 1.6Ghz quad core and Cortex-A7 quad core CPU	700 MHz ARM	Need high processing power to handle the four cameras streaming
GPU	5	GeForce 610M	PowerVR SGX544MP3 GPU (OpenGL ES 2.0, OpenGL ES 1.1 and OpenCL 1.1 EP	Broadcom VideoCore IV @ 250 MHz, OpenGL ES 2.0, MPREG-2 and VC-1 0180p	It is necessary for approximately four (4) cameras to be recording and processed. Therefore, video quality is important
Memory & Storage	4	4GB DDR3 1600/1333 320GB SATA Hard Drive (not included)	2GB LPDDR3 RAM PoP,eMMC 4.5 Flash Storage, Micro-SD slot	512MB (shared with GPU), SD slot	A reasonable size of data storage is necessary to store the compressed video files
Video Features	1	HDMI, VGA	HDMI 1.4a output Type-D connector	IN- CSI Input Connector, OUT - RCA, HDMI	Video features are of low priority because the HDMI/VGA is only needed in development
Audio Features	0	1 x S/PDIF out(Audio jack)	-	OUT - 3.5mm, HDMI	No audio is needed because it will be collected from the webcam
Software (Operating System)	3	Windows	u-boot 2012.7, Kernel 3.4.x, Android 4.2.x,	BSD and Linux	Linux is a preferred choice of OS because of the ease of use for programming

			Linux BSP will be supported Q4 2013.		development and short boot time
Power	5	120V wall outlet	DC 5V/4A	DC 5V	Low power is a high priority. The computer will be running off a battery in some cases
Peripherals	5	HDMI, VGA, 2x USB 3.0 4x USB 2.0	USB 3.0 Host x 1, USB 3.0 OTG x 1, USB 2.0 Host x 4	Ethernet Port, USB port	A USB port is needed for each webcam. The GPS module also needs a USB port. USB 3.0 is preferred due to the higher bandwidth than USB 2.0
Cost	4	\$201	\$169	\$59	Cost is a factor for budget constraints. High processing power, multiple USB ports, and low cost

****For All Gant Charts referenced in the document, see the attachments included.**